User Manual v.1

Function Library- MPC07

[This manual states all function library of MPC07]

Beijing Hengyuan Zhonglina Instruments Co., Ltd.

2011-01-01

CONTENTS

1. Development of Motion Control System

1.1 Development of motion control system based on Windows

With the Dynamic Link Library (DLL) of MPC 07, developers could develop the motion control system based on Windows rapidly. The Dynamic Link Library (DLL) of MPC 07 is the standard 32 digits DLL of Windows, and the development tool chosen by MPC07 should support standard 32 digits DLL quotation of Windows.

How to use Microsoft Visual Basic and Microsoft Visual C++ to develop the motion control system based on Windows is introduced as follows:

1.1.1 Visual Basic Control System Development

(1)Summary

To develop a motion control system based on Windows, users could use VB5.0 or a higher version. It's quite easy to develop a simple Visual Basic control program. According to the following steps, a simple control system would be developed rapidly.

1. Install MPC 07 driver program and function library;
2. Use Visual Basic to write an interface program;
3. Add MPC 07.bas file to VB Project;
4. Quote motion function in application program.

All the teaching materials of Visual Basic have introductions of how to write interface programs, including buttons, dialog box, menu, etc. For a developer who is familiar with Visual Basic and the motion functions library of MPC 07, a simple motion program which is based on Windows and composed of input boxes and command buttons could be developed within a few minutes.

（2）Quote method of the Dynamic Link Library function

To quote the functions of the Dynamic Link Library (DLL) in VB should include two parts as follows:

▌Functions statement

The statement of each function of Dynamic Link Library (DLL) in VB is included in the file of MPC 07.bas. This file could be found in the installation catalogue '\MPC07SP\Develop\VB' of the MPC07 board application. What is need user to do is just add the file to the VB project.

▌Functions quote

If the return value of the quote function is vacant or not need the return value, user could quote function with the method as follows:

```
con_pmove1,2000
```

or

Call con_pmove (1,2000)

If the return value is needed, user could quote function with the method as follows:

Dim rtn As Long

rtn=con_pmove(1,2000)

Note: the data type of transferred parameter and variable type of the received return value should consist with that of function statement, furthermore, all the parameters of int type(the integer type in C language) & long type(the long-integer type in C language) and return values in the functions description should adopt the Long data type (Long-integer type in VB); all the float (single floating-point type in C language) and double (double floating-point type in C language) parameters and return values should use Double data type(double floating-point type in VB), otherwise, there would be unexpected results.

(3) Use of demonstrate program

In the file of installation catalogue of MPC 07 board application "\MPC07SP\Demo\VBDemo", there are two motion control system demonstration demo program which are developed based on VB 6.0. Users could compile and run this demonstration according to the following procedures; after have acquainted with the corresponding programming method, users could develop the motion control system base on their own demand.

1. Install correctly according to the installation procedures of MPC 07.
2. Set up a new folder in the hard disc.
3. Copy all the files in the folder of installation catalog of MPC 07 board application "\MPC 07SP\Demo\VBDemo\Demo1" (or another demonstration program folder) to the new folder in the hard disk.
4. Start up VB6.0 integration environment and open project.
5. Make sure that the board card has been set correctly and inserted into computer.
6. Compile this project and create exe file.
7. Run the exe file.

1.1.2 Use Visual C++ to Develop Control System

（1）Development Environment

Users can use VC5.0 or a higher version to develop motion control system based on Windows.

（2）Method of quoting the functions in the dynamic link library

There are two methods of quoting the functions of the dynamic link library in VC

- Dim Quote method

Dim quote need the following files:

1. the head-files statement of DLL function MPC.h;
2. guide-in library file MPC 07.lib used in connection for compiling;
3. Dynamic Link Library files MPC 07.dll;
4. Device driver MPC 07.sys;

In the above files, (1) and (2) can be found in the installation catalogue of the MPC 07 board application "\MPC 07SP\Develop\VC"; (3) has been installed in C:\WINDOWS\SYSTEM32; (4) has been installed in C:\WINDOWS\SYSTEM32\DRIVERS (suppose that Windows has been Installed in the file C:\WINDOWS).

After setting up a project, click the menu "/project/settings…" in VC integration environment, "project settings" dialogue box would be popped up. Select "Link" options, input file name of guide-in library MPC 07.lib in "object/library modules", then Click "OK". MPC.h header file is included in the beginning of original file in quoting DLL function, then DLL functions could be quoted as quoting the internal functions. The detail could refer to demonstration: \Demo\VCDemo\Demo1.

▌ Explicit Quote

Explicit quote need the following files:

1. Dynamic Link Library file MPC 07.dll;
2. Device driver MPC 07.sys.

In the above files, (1) has been installed in C:\WINDOWS\SYSTEM32, (2) has been installed in C:\WINDOWS\SYSTEM32\DRIVERS (suppose Windows has been installed in the file C:\WINDOWS).

Explicit quote need to quote Windows API function load and release dynamic link library. The method is as follows:

1. Quote Windows API function LoadLibrary (), load the DLL dynamic;
2. Quote Windows API function GetProcAddress (), get the finger of function which will be quoted in DLL;
3. Use the finger of function to quote DLL function and complete corresponding function;
4. When the program end or there is no need for DLL functions, quote Windows API function FreeLibrary () and release dynamic link library.

This method is quite complicated. MPC 07 has already encapsulated the usual use DLL functions in MPC 07.dll into class CLoadDll and provide the original code of this class. This class has member functions which have the same function name and parameter with motion instruction library. The original code can be found in installation catalogue of MPC 07 board application "\MPC 07SP\ Develop\VC" with a file name of LoadDll.cpp and LoadDll.h. Developers can add it into the project; add the object of this class in the program properly, then quote DLL function according

to corresponding member functions. Refer to demonstration "\Demo\VCDemo\Demo2" for details.

Two methods above mentioned are standard methods for quoting DLL function in VC. Getting more detailed quoting methods and help, please refer to the corresponding contents of Microsoft Visual Stutio development documents MSDN, or relevant VC reference books.

（3）Use of demonstration program

Installation catalogue of MPC07 board application "\MPC 07SP\Demo\VCDemo\" has three motion control system demonstration Demo program which is developed in VC6.0. \Demo\VCDemo\Demo1 is a dim quote example, and \Demo\VCDemo\Demo2 is explicit quote example. Users can compile and run the demonstrations as the following procedures; after having acquainted with corresponding programming method, users could develop the dynamic control system according to their own demand.

1. Install correctly according to the installation procedures of MPC 07.
2. Set up a new folder in the hard disc.
3. Copy all the documents in Demo1 or Demo2 of installation catalog of MPC 07 board application "\MPC 07SP\Demo\VBDemo\Demo" to the new folder in the hard disk.
4. Start up VC 6.0 integration environment and open project demo1.dsw or demo2.dsw.
5. Make sure board has been installed correctly and inserted into computer.
6. Compile this project and create .exe file.
7. Run the .exe file.

In addition, the folder "\Demo\VCDemo\" has offered a MPC 07 function testing program "\Demo\VCDemo\Demo3", which is just an executable file to test all the MPC 07 functions.

2. Function Descriptions

This chapter describes each function of MPC 07 motion library in details. The unit and function return value used in function library is established by usage as follows:

Unit

●Unit of displacement (or distance) is P (Pulse);

●Unit of speed is PPS（Pulse/sec）;

●Unit of acceleration and deceleration is PPSS（Pulse/sec^2）.

Function return value

Most of the functions in motion library are integral type functions; generally, the meanings of theirs return value is as follows:

'0' Function performs correctly;

'-1' Function performs incorrectly.

2.1 Controlling Card and Axis Set Function

This type of function is mainly used for setting the number of MPC 07 card ,the number of controlling axis and output mode of each axis, setting and reading the speed & acceleration, etc. The relevant functions are as follows:

int auto_set (void); /* Auto-testing & auto-setting controlling card*/

int init_board（void）; /* Initialize the hardware and software of controlling card */

int get_max_axe（void）; /* Read total axes number */

int get_board_num（void）; /* Read the number of board card */

int get_axe（int board_no）; /* Read the axis number on board card */

int set_outmode（int ch，int mode，int outlogic）; /* Set output mode of axis*/

int set_conspeed（int ch，double conspeed）; /* Set constant velocity of axis */

double get_conspeed（int ch）; /* Read the constant velocity of the set axis */

int set_profile（int ch，double ls，double hs，double acc）; /* Set axis trapezoidal velocity */

int get_profile（int ch，double &ls，double &hs，double &acc）; /* Read the trapezoidal velocity of the set axis */

int set_vector_conspeed（double con_speed）; /* Set constant vector velocity */

int set_vector_profile（double vec_fl，double vec_fh，double vec_ad）;

/* Set trapezoidal vector velocity */

double get_vector_conspeed（void）; /* Read the set constant vector velocity */

int get_vector_profile（double *vec_fl，double *vec_fh，double *vec_ad）; /* Read the set trapezoidal vector velocity */

double get_rate（int ch）; /* read the Current actual velocity of axis*/

Function Name: auto_set

Purpose: use auto_set to test the number of MPC 07 board, number of axis of each board automatically, and auto set each MPC 07 controlling card.

Syntax: int auto_set（void）;

Quoted Examples: auto_set（）; /* Auto-testing & auto-setting controlling card*/

Description: auto_set is used to test the number of board, axis, and set these parameters

automatically. This function can only be quoted once in the program.

Return Value: if successfully quoted, auto set function return the total axis number; if no board can be tested, return '0' value, if failed to quote, then return negative value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: init_board

Purpose: Use init_board function to initialize the controlling card.

Syntax: int init_board（void）;

Quoted Examples: init_board（）;

Description: after auto-testing and auto-setting with auto_set function, init board function has to be used to initialize controlling card. Init_ board function is mainly used to initialize each register of the controlling card, impulse output mode of each axis(impulse/direction),constant velocity（2000pps）,trapezoidal velocity(beginning speed 2000pps, high speed 8000pps，acceleration and deceleration speed 80000ppss),constant vector velocity, vector trapezoidal velocity (beginning speed 2000pps, high speed 8000pps，acceleration and deceleration speed 800 00ppss), etc. This function can only be used once.

Return Value: if successfully quoted, init_board function return the numbers of inserted board card; if no board can be tested, return to '0'value, if return to negative value, it means that there is wrong.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Note: if not quote init_ board function to initialize, controlling card could not work properly. If change initial data of impulse output mode, speed, etc, other functions could be quoted.

Reference: auto_set

Function Name: get_max_axe

Purpose: get_max_axe function is used to read total number of controlling axis.

Syntax: int get_max_axe（void）;

Quoted Examples: max_axe_num=get_max_axe（）;

Return Value: get_ max_ axe function returns the total number of controlling axis.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: get_board_num

Purpose: get_board_num function is used to read total number of the installed board cards.

Syntax: int get_board_num（void）;

Quoted Examples: card_num=get_board_num（）;

Return Value: get_board_num function returns to total number of board card.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: get_axe

Purpose: get_axe function is used to read total axis number of board card.

Syntax: int get_axe（int board_no）;

board_no: card number;

Quoted Examples: axe_num=get_axe（1）;

Return Value: get_board_num function returns to total number of board card.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: set_outmode

Purpose: set_ outmode function is used to set impulse output mode of each axis. If driver demand double impulse (positive impulse, negative impulse) control signal interface, then this function should be quoted after init_board function.

Syntax: int set_outmode（int ch，int mode，int outlogic）;

ch: controlling axis of the set output mode;

mode: setting of impulse output mode (1 for impulse/direction mode, 0 for double impulse mode)

outlogic: this parameter is invalid in MPC 07, and it can be set any value.

Quoted Examples: set_outmode（2，0，1）; /* set the impulse output mode of the 2nd axis double impulse mode. */

Description: on condition of default, init_board function set all the axes impulse/direction mode. If driver demand double impulse (positive impulse and negative impulse) input mode, set_outmode function should be quoted after init_board function to reset the demanded mode. Notes: output mode of controlling card should be the conform to the input signal mode of the connected driver; otherwise, motor could not work normally.

Return Value: if output mode is set successfully, return value of set_outmode function is' 0'value; if not, it is '-1'value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: init_board

Function Name: set_maxspeed

Purpose: set_maxspeed function is used to set the maximum velocity of each axis.

Syntax: int set_maxspeed（int ch，double speed）

ch: the set controlling axis

speed: the set maximum velocity, its unit is impulse/second （pps）

Quoted Example: set_maxspeed（2，10000） /* set the maximum velocity of 2nd axis for 10000pps */

Description: on the condition of default, init _board function will set all axes for the maximum velocity the board allowed. To get better accuracy, it can be set according to actual output velocity when using. MPC 07 card output impulse frequency is controlled by two variables: impulse resolution and ratio, the multiply result of the two variables is output impulse frequency. Quote set_ maxspeed function to set the maximum impulse frequency demanded, and then impulse resolution would be reset.

Return value: if output mode is set successfully, set_ maxspeed function return to '0' value, if not, return negative value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: refer to "how to improve velocity accuracy" in Chapter 6.

Function Name: set_conspeed，get_conspeed

Purpose: set_conspeed function is used to set a constant velocity of an axis; get_conspeed function is used to get the constant velocity of set for an axis.

Syntax: int set_conspeed（int ch，double conspeed）;

double get_conspeed（int ch）;

ch: serial number of controlling axis

conspeed: the set constant velocity, its unit is impulse/second （pps）

Quoted Example: set conspeed (2,4 0 0)

speed=get conspeed(2);

Description: Function set_conspeed can be set the speed under the condition of regular speed. If the function called multiply, the value which is set last time is always effective until it is changed next time.

Return value: If constant speed value was set successfully, function set_conspeed return '0' value, or else return negative number.

get_conspeed function return constant velocity value of the set axis or else return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Notes: Constant velocity value is generally low to avoid controlling motor (especially the open-loop stepper motor) losing step or overshooting. It's best to use trapezoidal speed mode in

high-speed movement condition.

Refernce: set_profile, set_vector_conspeed

Function Name: set_profile, get_profile

Purpose: set_profile function is used to set parameters value of trapezoidal speed under fast movement condition (including fast_hmove, fast_vmove, fast_pmove, etc). Get-profile is used to read trapezoidal speed parameter value.

Syntax: int set_profile（int ch,double ls,double hs,double accel）

ch: the number of controllling axis

ls: set low speed value (initial speed); unit: pps (pulse/second);

hs: set high speed value (target speed);unit: pps(pulse/second)

accel: set the acceleration; unit: ppss(pulse/second/second)

int get_profile（int ch,int *ls,int *hs,long*accel）

double *ls: point to the finger of initial speed

double *hs: point to the finger of target speed

double *accel: point to the finger of acceleration

Quoted Example: set_profile（3,600,6000,10000）;

get_profile,（3,&ls,&hs,&accel）;

Description: set_profile function can be used to set low speed (initial speed), high speed (target speed), and acceleration value /deceleration velocity of an axis (deceleration velocity value equal to acceleration value) under fast movement condition. The default values of these parameters are 2000,8000,80000.

get _profile function return low speed value, high speed value and acceleration or deceleration speed value of trapezoidal speed set for an axis through finger.

Return Value: If set parameter value successfully, set_profile function return '0' value, or else return negative number.

If quoted successfully, get_profile return '0 'value, or else return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP.

Reference: set_conspeed, set_vector_conspeed, set_vector_profile

Function Name: set_vector_conspeed, get_vector_conspeed

Purpose: set_vector_conspeed function is used to set vector velocity which is used in two or three axis linear interpolation movement under constant speed condition. Get_vector_conspeed function can be used to read vector velocity under constant-speed condition.

Syntax: int set_vector_conspeed（double vec_conspeed）;

vec_conspeed: Vector velocity during constant speed interpolation.

double get_vector_conspeed（void）;

Quoted Example: set_vector_conspeed（1000）;

vec_conspeed= get_vector_conspeed（）

Description: Set_vector_ conspeed function set vector velocity for two or three axes interpolation constant speed movement function, such as con_line2, con_line3, but not for high speed interpolation movement such as fast_lin2, fast_line3 (which velocity is decided by set_vector_profile function). Get_vector_conspeed function returns constant vector speed. Constant vector speed called last time by the fouction of set_vector_conspeed is effective.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP.

Notes: Vector speed should be set relatively slower to avoid losing steps in process of exercise. It is available for movement interpolation at high speed condition such as fast_line2, fast_line3 to set movement speed with set_vector_profile function.

Reference: set_vector_profile, set_conspeed, set_profile

Function Name: set_vector_profile, get_vector_profile

Purpose: set_vector_profile function is used to set parameter of vector trapezoidal speed under the fast movement condition. Get_vector_profile function can be used to get vector trapezoidal speed parameter value under fast movement condition.

Syntax: int set_vector_profile（double vec_fl,double vec_fh,double vec_ad）;

vec_fl: Speed value with low speed of vector;

vec_fh: Speed value with high speed of vector;

vec_ad: Acceleration value with high speed of vector;

int get_vector_profile（double *vec_fl,double *vec_fh,double *vec_ad）;

*vec_fl: the finger pointing to low speed of vector

*vec_fh: the finger pointing to high speed of vector

*vec_ad: the finger pointing to acceleration of vector

Quoted example: set_vector_profile（1000,16000,10000）;

get_vector_profile（&vec_fl,&vec_fh,&vec_ad）;

Description: set_vector_profile function is used to set trapezoidal vector speed. However, it can not be used to set movement speed for con_line2, con_line3 function.

Return Value: If called succeed, set_vector_profile function and get_vector_profile funcitonn return '0' value, otherwise return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP.

Reference: set_vector_conspeed, fast_line2, fast_line3

Function Name: get_rate

Purpose: get actual rate of certain axes with get_rate function

Syntax: double get_rate（int ch）

ch: set controlling axes number.

Quoted Example: speed=get_rate（2）;

Description: get_rate function is used to read the current actual speed of the controlling axes. It is possible that the actual rate read by the function makes great differences with pulse rate set by the function set_conspeed and set_profile, which dues to the differences caused by controlling card speed resolution. Output pulse frequency of MPC 07 is controlled by two variables: pulse frequency and magnification, and the multiply result of the two variables is output impulse frequency. The maximum output pulse frequency set by the set_max speed function is revised pulse resolution; the actual output speed can be set in accordance with the actual maximum output speed to get better speed of accuracy.

Return Value: get_rate function return the current move speed of set axis and the unit is (pps).If called failed, function return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP.

Reference: 'How to Improve the Accuracy Speed' in chapter 6.

2.2 Movement Instruction Function

There are three movement types: point movement, continuous movement and return-origin movement; two movement ways: independent movement and interpolation movement; while two move speed movement: constant and fast speed (trapezoid). In order to describe conveniently, the movement instruction can be divided into independent movement and interpolation movement.

2.2.1 Independent Movement Function

The independent movement refers to no linkage relation among movements of each controlling axis, which is either uniaxial movement or multi-axial movement according to respective speed at the same time. Point movement, continuous movement and return-origin movement belong to independent movement. The command format of independent instruction movement function is X_YmoveZ.

The above-mentioned:

X: Substituted by con and fast, con is constant movement, fast is fast movement

Y: Substituted by p, v and h,'p' is point movement,'v 'is continuous movement and 'h' is return-origin movement.

move: Main body of instruction, indicating the instruction for movement instruction

Z: When the number is vacant, it is uniaxial movement, 2 for two axes independent movement while 3 for three-axle independent movement.

Example: con_vmove function is uniaxial constant continuous movement; con_pmove2 function is two-axis of constant point movement while fast_hmove3 is three-axis of fast return-origin movement instruction. For constant movement instruction, movement speed is set by set_conspeed function, while quick movement instruction the speed is set by set_profile functions. Take point movement, continuous movement and return origin movement for example to respectively illustrate the meaning of movement instructions.

(1)Point Movement Function

Point movement means the controlled axis move a set distances with their own speed and stop automatically when reach to target position. Notes: each axis start to move at the same time while not necessarily reach to target position simultaneously in two axes or three axes point movement function. The MPC 07 function library provides six command functions of movement command.

int con_pmove（int ch,long step）;/*one axis makes point movement with constant speed.*/

int fast_pmove（int ch,long step）;/*one axis makes point movement with fast speed*/

int con_pmove2（int ch1,long step1,int ch2,long step2）;/*two axes makes point movement with constant speed*/

int fast_pmove2（int ch1,long step1,int ch2,long step2）;/*two axes makes point movement with fast speed*/

int con_pmove3（int ch1,long step1,int ch2,long step2,int ch3,long step3;/*three axes makes point

movement with constant speed*/

int fast_pmove3（int ch1,long step1,int ch2,long step2,int ch3,long step3）;/*three axes makes point movement with fast speed*/

The above-mentioned:

ch,ch1,ch2,ch3: Number of the Controlled axis

Step, step1, step2, step3: The distances of the controlled axis start to move from the current position. The positive number means positive direction while the negative number means the negative direction

Quoted Example:

con_pmove（1,-2000）; / *the first axis moves a distance of 2000pulses to the negative direction with the constant speed*/

fast_pmove2（2,5000,3,-1000）/ *The second axis moves a distance of 5000 pulses to the positive direction with the fast speed; The third axis moves a distance of 1000pulses to the negative direction with the fast speed */

Return Value: If quoted successfully, these functions return '0'value, else return '1' value.

(2)Continuous Movement Function

Continuous movement means the controlled axes move to the set directions with theirs own speed, and do not stop until touch the limit switch, alarm switch, or called the breaking function. MPC 07 function library provides six continuous movement command functions:

int con_vmove（int ch,int dir）;/*one axis moves continuously with constant speed*/

int fast_vmove（int ch,int dir）;/* one axis moves continuously with fast speed */

int con_vmove2（int ch1,int dir1,int ch2,int dir2）;/* two axes move continuously with constant speed */

int fast_vmove2（int ch1,int dir1,int ch2,int dir2）;/* two axes move continuously with fast speed */

int con_vmove3（int ch1,int dir1,int ch2,int dir2,int ch3,int dir3） /* three axes move continuously with constant speed */

int fast_vmove3（int ch1,int dir1,int ch2,int dir2,int ch3,int dir3）;/* three axes move continuously with fast speed */

The above-mentioned:

ch,ch1,ch2,ch3:No. of the controlled axes;

dir,dir1,dir2,dir3: Means the movement directions of the controlled axes; '1'means positive direction;''-1''means negative direction.

Quoted Example: con_vmove（1,'-1'）;/* the first axis moves continuously to the negative direction with the constant speed */

fast_vmove2（2,1,3,'-1'）;/* the second axis moves continuously to the positive direction with the rapid speed; the third axis moves continuously to the negative direction with the fast speed */

Return Value: If called successfully, these functions return '0' value, else return '1' value.

(3) Return-Origin Function

Return-Origin movement means the controlled axis move continuously to the set directions with theirs own speed, and do not stop until touch the limit switch, alarm switch, or called the brake function. MPC07 function library provides six return-origin movement command functions:

int con_hmove（int ch,int dir）;/*return origin with the constant speed */
int fast_hmove（int ch,int dir）;/* Return origin with the fast speed */
int con_hmove2（int ch1,int dir1,int ch2,int dir2）;

/*Two axises return theirs own origin with constant speed */

int fast_hmove2（int ch1,int dir1,int ch2,int dir2）;
/* Two axises return theirs own origin with fast speed */
int con_hmove3（int ch1,int dir1,int ch2,int dir2,int ch3,int dir3）;
/* Three axes return theirs own origins with constant speed */

int fast_hmove3（int ch1,int dir1,int ch2,int dir2,int ch3,int dir3）;
/* Three axises return theirs own origins with fast speed */
The above-mentioned:
ch,ch1,ch2,ch3: No. of the controlled axes;
dir,dir1,dir2,dir3: Means the movement direction of the controlled axes .1 means positive direction: '-1' means negative direction

Quoted Example:

con_hmove（1,'-1'）;/*the first axis make return-origin movement to the negative direction with constant speed*/
fast_hmove2（2,1,3,'-1'）;/*the second axis make return-origin movement to the positive direction with rapid speed; the third axis make return-origin movement to the negative direction with rapid speed. */

Return Value: if called successfully, these functions return '0' value, otherwise return '-1' value.

Notes: For V1.0 version movement control card of MPC 07SP, the external origin switch limit switch and alarm switch may be either normally opened type switch or normally closed type switch. Interface function of set_org_logic,set_el_logic,set_alm_logic can be used to set the effective level.

2.2.2 Interpolation Movement Function

Interpolation movement refers that two or three axes carry on linkage according to certain algorithm and the controlled axis start at the same time while reaching target position simultaneously. Interpolation movement operates with vector speed. Vector speed is divided into constant vector speed and trapezoidal vector speed. The functions related to interpolation movement are as follow:

Linear interpolation function refers that two or three axes make linear linkage with vector speed

(constant vector speed or trapezoidal speed), Running speed of each controlled axis is velocity component of vector speed in the axis. Controlled axes start at the same time while reaching the target position simultaneously. MPC07 function library provides four linear interpolation functions.

int con_line2（int ch1,long pos1,int ch2,long pos2）;/*two axis make linear movement with constant speed */

int fast_line2（int ch1,long pos1,int ch2,long pos3）;/*two axis make linear movement with rapid speed*/

int con_line3（int ch1,long pos1,int ch2,long pos2,int ch3,long pos3）;/* three axis make linear movement */

int fast_line3（int ch1,long pos1,int ch2,long pos2,int ch3,long pos3）;/*three axis make linear movement with rapid speed. */

The above-mentioned:

ch1,ch2,ch3:No. of the controlled axis.

pos1,pos2,pos3: means the distance of the controlled axis start to move from the current position, the positive number means the positive direction while the negative number means the negative direction. Its unit is the number of pulse.

Quoted Example:

con_line2（1,-2000,3,1000）;/*the first and third axis make the linear interpolation movement with constant vector speed. The first axis moves a distance of 2000 pulses to the negative direction while the third axis moves a distance of 1000 pulses to the positive direction.

fast_line3（2,5000,3,1000,5,3000）;/* the second, third and fifth axis make the linear interpolation movement with trapezoidal vector speed. The second axis moves a distance of 5000 pulses to the positive direction; the third axis moves a distance of 1000 pulses to the negative direction while the fifth axis moves a distance of 3000 pulses to the positive direction. */

Return value: if called successfully, these function return '0'value, otherwise return to '-1' value.

2.3 Brake Function

MPC 07 movement controlling operation manual can be called if it is necessary to pause or stop an axis or more in operation process. Six braking function exist in the MPC 07 movement function library.

void sudden_stop（int ch）;*/ stop an moving axis suddenly*/

void sudden_stop2（int ch1，int ch2）;*/ stop two moving axis suddenly */

void sudden_stop3（int ch1，int ch2，int ch3）;*/ stop three moving axis suddenly */

void decel_stop（int ch）;/* stop one moving axis smoothly */

void decel_stop2（int ch1，int ch2）;/* two moving axis smoothly*/

void decel_stop3（int ch1，int ch2，int ch3）;/* stop three moving axis smoothly */

including：

ch,ch1,ch2,ch3：No. of Controlled axis.

Quoted Examples：decel_stop（2）;/* stop second moving axis smoothly */

sudden _stop2（1，4）;/* stop first and forth axis suddenly*/ decel_stop3（1，2，3）;/* stop first, second and third axis smoothly */

Return Value：If called succeed, return 0, or else return '-1' or the unbraked axis no.

Notes：decel _stop function is available to all types of motion. Decel _stop is for （fast_YmoveZ）, it can slow down the controlled axis from high speed(set up by set_profile), and then stop. In general, Decel function can be called to stop fast movement smoothly (such as, fast_hmove, fast_vmove, fast_pmove2) to avoid overfast.

The controlled axis stop suddenly when sudden_stop function is called. After sudden_stop carried out, controlling card stop sending pulse to motor driver immediately to make it stop. Generally, this function called in a emergency stop. Decel_stop can not be called in the common movement mode.

2.4 Position and Status Set Function

int set_abs_pos (int ch，long pos) ;/* set the absolute position of a axis*/

int reset_pos（int ch）;/* reset the current position value as '0' */ int reset_cmd_counter (); /* used to reset the counter of motion command */

int set_dir(int ch，int dir);/* set moving direction of a axis */

int enable_sd(int ch，int flag);/*set external slow down signal of a axis as valid or not*/

int set_sd_logic（int ch，int flag）;/*set external slow down signal valid level */

int set_el_logic（int ch，int flag）;/*set external limit signal valid level of a axis */

int set_org_logic（int ch，int flag）;/*set external original signal valid level of a axis */

int set_alm_logic（int ch，int flag）;/* set external alarm signal valid level of a axis */

Function Name：**set_abs_pos**

Purpose：It used to set the absolute start position of the axis movement.

Syntax：int set_abs_pos（int ch，long pos）;

ch：No of the Controlling axis.

pos：absolute position(current position) of start of the Controlling axis

Quoted Example：set_abs_pos（1，1 0 0 0）;/*set the first axis current position as 1000*/

Descriptions:

To quote the function could set a value for the current position. But for the axis, there is no actual movement from the last position to current position. Quote the function, and set the second parameter ' 0' value, which could carry out the function of the reset_ pos() function.

Return Value: If quoted successfully, these functions return ' 0' value, else return '1' value.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

Function Name: reset_ pos

Syntax: int reset_ pos（int ch）

ch: the serial numbers of the controlling axis

Quoted Examples: int reset_ pos（1）

Description: Reset _pos function reset the absolute position and relative position of the designate Axis 0, usually, it is quoted when the origin is found. The value of the current position must be 0, from then, all of the absolute position is relative to this position. The axis must be stop when The function is quoted, otherwise, which may cause the confusion of the absolute positions.

Return Value: If quoted successfully, reset_ pos functions return 0value, else return '-1'value.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

Notes: Generally, this function should be quoted after carring out the con_hmove function or fast_hmove function successfully.

Reference: get_abs_pos ,get_rel_pos

Function Name: reset_cmd_counter

Purpose: It is used to take count of the movement command and reset the command 0value.

Syntax: int reset_cmd_counter();

Quoted Examples: reset_cmd_counter();

Description: After the initialization of the movement command, it is the beginning to take count of It. Along with the movement command (be able to give birth to command of movement, excluding the setting command such as set_conspeed function) carring out increase by degrees, it is normal to quote reset_cmd_counter() function to reset the ' 0' value.

Return Value: If quoted successfully, reset_ pos functions return '0' value, else return '-1' value.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

Reference: set_cmd_counter

Function Name: set_dir

Purpose: It is used to set the direction of the movement of the axis.

Syntax: int set_dir(int ch, int dir)

ch: the axis be going to be set

dir: means the direction of the movement of the axis be controlled. '+1' means the positive direction, '-1' means the negative direction.

Quoted Examples: set_dir(1,'-1'); /*set the direction of the movement of the first axis negative direction*/

Description: Before the new movement command be given, to quote the function could set the direction of the movement of certain axis.

Return Value: If quoted successfully, reset_ pos functions return '0'value, else return '-1' value.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

Function Name：enable_sd

Purpose：It used to set an external slow-down signal of a axis valid or not.

Syntax：int enable_sd（int ch，int flag）;

ch：No. of the Controlling axis.

flag：Flag is the signal to identify the outside deceleration is valid or not.'1' for valid; ' 0' for invalid.

Quoted Example：enable_sd（1，0）;/*set the external decelerate signal of the first axis invalid. */

This function is called to set the external deceleration signal of an axis is valid or not. If the external decelerate signal of an axis is set invalid, the corresponding decelerate signal input(SD4～SD1) can be used as general input. Ceckin_byte，checkin_bit can be used to read the status of the corresponding port. In this way, general input port can be increased to 2 0. For SD4～SD1,the corresponding general ports are followed by 20～17.

Return Value：If called succeed, enable_sd return '0' value, or else return '-1' value.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference：checkin_byte，checkin_bit

Function Name：set_sd_logic

Purpose：set_sd_logic is used to set external slow-down signal valid level of an axis.

Syntax：int set_sd_logic（int ch，int flag）;

ch：No. of Controlling axis.

flag：Flag is the external slow down signal valid level. '1' means external decelerate switch trigger at high level; '0' means external decelerate switch trigger at low level.

Quoted Example：set_sd_logic（1，1）;/* set external slow-down signal high level when it is valid*/

Description: Call this function to set validity level produced by control axis deceleration. If it is validity to set the outer deceleration of control axis for high level, and the corresponding input deceleration signal is high level, the axis will decelerate automatically. When initialization, the default value is low level.

Function Descriptions:

Return Value：If called succeed, Set_sd_logic return '0' value, or else return '-1' value.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name：set_el_logic

Purpose：set_el_logic function is used to set the external limit signal valid level of an axis

Syntax：int set_el_logic（int ch，int flag）;

ch：No. of Controlling axis.

flag：Flag is the sign of valid level of outside limit signal. '1' means the external limit switch trigger at high level; '0' means the external limit switch trigger at high level.

Quoted examples：set_el_logic（1，1）/* set external limit signal of the first axis valid when it is at the high level.

Function Descriptions

Description：This function is called to set the valid level of external limit signal of the controlling axis. If the external limit signal of controlling axis was set valid at the high level, when the input port of limit signal is high level at a certain direction, the axis automatically stop in that direction. When initialization, the default value is low level

Return Value：If called succeed, the function return '0'value, or else return '-1'value.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name：set_org_logic

Purpose：It is used to set valid level of the external original signal of an axis.

Syntax ：int set_org_logic（int ch，int flag）;

ch: No. of controlling axis.

flag：It is the sign of the valid level of external original signal. '1' means the external original switch trigger the control card at the high level; '0' means the external original switch trigger the control card at the low level.

Quoted Examples ：set_org_logic（1，1）;/* set external original signal of the first axis valid at the high level*/

Description：set_org_logic can be called to set the valid level of the external original signal of the controlling axis. If the external original signal of the controlling axis is set high level valid, and the corresponding original signal input port is high level which means the axis return original.

When initialization, the default value is low level

Return Value：If this function is called succeed, the return value is '0'; or else is '-1'.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name：**set_alm_logic**

Purpose：**set_alm_logic** is used to set the valid level of external alarm signal of a axis.

Syntax：int set_alm_logic（int ch，int flag）;

ch：No. of controlling axis.

flag：Flag is the symbol of valid level of external alarm. '1' means the external switch trigger the control card at the high level; '0' means external switch trigger the control card at the low level.

Quoted Example：set_alm_logic（1，1）;/* set external alarm signal of the first axis valid at the high level */

Description: MPC 07 motion controlling card has a public alarm signal. Thus, if the valid level of the external alarm signal of any axis is set ,other axis can be set in the same. Set_alm_logic function can be called to set the valid level of external alarm signal. If the the valid level of external alarm signal is set high level, all the axis automatically stop when the corresponding alarm signal input port are high level.

Return Value：If called succeed, return '0' value; or else return'-1' value.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

2.5 Positions& Status Function

In the motion curse, position or status Query Function could be called if need the motion position or condition of a certain axis.

2.5.1 Get-Position Function

int get_abs_pos（int ch，long*pos）;/*return the absolute position of a axis*/

int get_rel_pos（int ch，long*pos）;/* return the relative position of a axis */

int get_cur_dir (int ch);/*return the current moving direction of a axis*/

Function name: get_abs_pos, get_rel_pos

Purpose: get_abs_pos function is used to get the absolute position which is relative to the original position or the origin. get_rel_pos is used to get the relative position which is relative to the origin of the current movement.

Syntax: int get_abs_pos（int ch，long*pos）; int get_rel_pos（int ch，long*pos）;

ch: the serial numbers of the controlling axes.

abs_pos: a long-integer finger which point to the absolute position

rel_pos: a long-integer finger which point to the relative position

Quoted Examples:

temp=get_abs_pos(1,&abs_pos);temp=get_rel_pos(1,&rel_pos);

Description: get_abs_pos function is used to get the current absolute position of the controlling

axes.This absolute position is relative to the origin if the function has carried out the movement of returning to the origin; if not, the absolute position is relative to the position when opening. Get_rel_pos function is used to get the relative position which is relative to the original position of current movement. If the designate axis does not move currently, the relative position of the axis is ' 0'. Because the positions got by the two functions are the quantity of the output pulses of the controlling-card, so the position can not reflect the actual position even if they lost or excessive-pulse.

Return Value: If quoted successfully, get_abs_pos functions and get_rel_pos function return '0'value, else return '1'value.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

Function Name: get_cur_dir

Purpose: get_cur_dir function is used to get the current direction of movement.

Syntax: int get_cur_dir (int ch)

ch: the serial numbers of the controlling axises

Quoted examples: get_cur_dir (1) /*get the current movement direction of the first axis*/

Return Value: If quoted unsuccessfully, get_cur_dir function return '-2'value; If return '1'value, which is means that the controlling axis is in the negative direction of movement; If return '1', that's means that the controlling axis is in the positive direction of movement; And the ' 0' value means the axis is stop.

System: WINDOW98, WINDOWS 2000, WINDOWS XP

2.5.2 Check Status Function

int check_status（int ch）;/* check status of an axis */
int check_done（int ch）;/* check a movement of an axis is done or not */
int check_limit（int ch）;/*check an axis reach the limit switch position or not */
int check_home(int ch);/*check an axis reach the original switch or not */
int check_SD(int ch); /* check an axis reach the decelerate switch position or not*/
int check_alarm(int ch);/*check an axis reach the alarm switch or not */
int get_cmd_counter(); /*used to get the current command counter */

Function Name：check_status

Purpose: check_status is used to read the current status of an axis

Syntax：int check_status（int ch）;

ch：No. of the Controlling axis.

Quoted Example：ch_status=check_status（2）;

Description: check_status function read status of the specified axis. Each axis of MPC 07 movement controlling card has one 32 bits status word (double). And thus the current working status of an axis can be checked. Each bit's meaning of the double status is as follows:

Return Value: If called succeed, check_status return the status value of the specified axis, or else return '-1'.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference：

D31
~
D27
D26 '0': No ORG signal; '1': have ORG origin switch and its condition
D25 '0': No 'EL+' signal; '1': means having EL+ positive limit switch and EL+ condition
D24 '0': No EL- signal; '1': have EL- negative limit switch and EL- condition
D23
~
D16
D15 0: no ALM signal;1: have ALM alarm signal
D14
D13 1: ALM signal; 0: Reasons and symbols of motion stop for non-ALM signals
D12
D11
D1 0 '1': ORG signal; '0': Symbols of motion pause for non-ORG signals
D9 '1': EL_P signal; '0': Symbols of motion pause for non-EL_P signals
D8 '1': signal EL_N; '0': Symbols of motion pause for non-EL_N signals
D7 '0': Pause;'1': The moving status of the current axis
D6
D5
D4
D3
D2
D1 '0': have SD signal;'1': No SD signal
D 0

Function Name: check_done
Purpose: Check_done is used to check the operation of the Specify Axis is done or not.
Syntax: int check_done (int ch);
ch: No. of Controlling Axis.
Description: Function check_done is used to check the working status of the specified axis.
Return Value: If the specify axis is in working status, check_done return' 1'value; if the specify axis is in static state, check_done return '0' value ;if function call is failed ,it return '-1'value.
System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: check_limit
Purpose: Check_limit function is used to check an axis reach the limited switch or not.
Syntax: int check_limit (int ch);
ch: No. of Controlling Axis.
Quoted Example: status=check_limit (1);
Description: For MPC 07 operation control card,each axis has two limit switch input port :positive limit signal input and negative signal input. Check_limit is used to check the sate of the specified axis ' limit switch, and return that the specified axis reach the limit switch or not, and to which direction.

Return Value：If check_limit return '1', which means reach the positive limit switch position; return '0' means did not reach the limit switch position; return '2' means reach the two limit switches; if failed called, return '-3'.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name：**check_home**

Purpose：Check_home function is used to check one axis reach origin switch or not.

Syntax：int check_home（int ch）;

ch：No. of Controlling Axis.

Quoted Example：status=check_home（1）;

Description ：For MPC 07 controlling card, each axis has a origin switch input port. Check_home is used to check the origin switch status of the specified axis, to check the specified axis reach the origin switch or not.

Return Value：If check_ hone return '1', it means reach the origin switch position; return '0' means did not reach the limit switch position; return '-3' means failed called.

System：WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name：**check_SD**

Purpose：Check _SD function is to check one axis reach slowdown switch position or not.

Syntax：int check_SD（int ch）;

ch：No. of Controlling Axis.

Description：Each axis of MPC 07 has a slow down switch input port. Check_SD is used to check decelerate switch status of the specified axis.

Return Value: If check_SD function returns to '1'value, it means the axis reaches the position of speed-down switch. Otherwise, return ' 0' value. If quoted incorrectly, it will return '-3' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function name: check_alarm

Purpose: check_alarm function is used to check an axis reaches the position of alarm switch or not.

Syntax: int check_alarm（int ch）;

ch：No. of Controlling Axis.

Description: All axes of motion controlling card share one alarm switch input . Check_alarm function is used to check the status of the alarm switch; and return that if there is validity alarm signal input board card.

Return Value: Check_alarm function returns '1'value means it reaches the position of alarm switch, if returns '0' value means it doesn't reaches the position of alarm switch, if quoted unsuccessfully, it will return '-3' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: get_cmd_counter

Purpose: this function is used to get command counter of the movement which is executed currently.

Syntax: int get_cmd_counter（）;

Quoted Example: cmdcounter=get_cmd_counter（）;/* read command counter of the movement

which is executed currently and reserve it in the variable of cmdcounter */

Description: This function can be used for querying the motor command is executing currently or not. Command counter of the movement start with 0, after completing initialization; increase with the executive motor command (that is command leads to movement, and it doesn't include setting commands such as set_conspeed); and it can quote reset_cmd_counter() function to clear.

Return value: If quoted successfully, the return value would be the value of command counter of the movement executed currently, otherwise return '1'value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: reset_cmd_counter

2.6 I/O Port Operation Fuction

int checkin_byte(int cardno); /*read states of all common input of controlling card*/
int checkin_bit(int cardno,int bitno);/*read the state of certain common input of controlling card*/
int output_byte(int cardno,int bytedata);/*write to all common output of controlling card*/
int output_bit(int cardno,int bitno,int status);/*write to certain common output of controlling card*/

Function Name: checkin_byte

Purpose: Read the state of switch volume of common input on board card.

Syntax: int checkin_byte(int cardno);

cardno: serial number of card

Description: MPC07 card provide sixteen common photoelectricity seclusion inputs which located on the extended board of EA1616 for users. The states of these sixteen inputs can be read by this function. Connection can be found in the section of MPC 07 Connector. If quote enable_sd Function to set the exterior speed-down signal invalid, common inputs could be increased to twenty , SD4～SD1 respectively correspond to input 20~ and input 17.

Return value: return the state of input, return value of 1~16(binary) correspond to 16 inputs. If the bit is '1', it means there is high electrical level input; if the bit is '0', it means there is low electrical level input; if an error appears,it return '1'.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: checkin_bit, enable_sd

Function Name: checkin_bit

Purpose: Read a certain switch volume states of the common input of board card.

Syntax: int checkin_bit(int cardno,int bitno);

cardno: serial number of card

bitno:Means certain switch volume to be read, with the value range of 1~20.

Description: MPC 07 card provide sixteen common photoelectricity seclusion inputs which located on the extended board of EA1616 for users. The states of a certain input can be read by this function. Connection can be found in the section of MPC 07 Connector. If quote enable_sd Function to set the external speed-down signal invalid, common inputs could be increased to twenty , SD4～SD1 respectively correspond to input 2 0~ and input 17.

Return Value: Return the state of certain input. '1' value means high electrical level input; '0' value

means low electrical level input; if an error appears, it return '1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: checkin_byte, enable_sd

Function Name: output_byte

Purpose: Set the switch volume states of common output of board card.

Syntax: int output_byte(int cardno,int bytedata);

cardno: serial number of card with the value range from1 to the maximum (card quantity).

bytedata:Information of the state of output needed to set.

Description: MPC 07 card provide 24 common photoelectricity seclusion inputs to users, among which common output 1～8 located on main MPC 07 board, the others located on the EA1616 extended board. The states of these 24 outputs can be set with the function. Connection can be found in the section of MPC 07 Connector. All bits of parameter 'bytedata' corresponding to the outputs one-to-one, and that means the lowest bit D1 corresponding to common output 1, the second bit D2 corresponding to 2 and followed by analogy.

Return Value: Return '0' if setting successfully, return '1' under the mistake.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: output_bit

Function Name: output_bit

Purpose: Set certain switch volume states of the output on the board card.

Syntax: int output_bit(int cardno,int bitno,int status);

cardno: Serial number of card with the value range from1 to the maximum(card number).

bitno: Means certain common output with the value range of 1~24

status: Setting status; 1: ON; 0: OFF

Description: MPC 07 card provide 24 common photoelectricity seclusion inputs to users, among which common output 1～8 located on main MPC 07 board, the others located on the EA1616 extended board. The status of certain output can be set with the function. Connection can be found in the section of MPC 07 Connector.

ReturnValue: return ' 0' if setting successfully, return '1' under the mistake.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: output_byte

2.7 Other Functions

int set_backlash（int ch, int backlash）/*Set compensation value of gaps caused by the direction turning of mechanism*/

int start_backlash（int ch）;/* Compensation start */

int end_backlash（int ch）;/* Compensation end*/

int change_speed(int ch,double speed); /*Speed-changing in movement */

int Output（int portid,unsigned char byte）;/*Output a byte to a certain address*/

int Input（int portid）;/*Input a byte at certain address*/

int get_sys_ver(long* major,long *minor1,long *minor2);

/* used for querying the version of function library*/

int get_sys_ver(long* major,long *minor1,long *minor2);
/ used for querying the version of driver/
int get_card_ver(long cardno,long *type,long* major,long;
/* used for querying the hardware version of motion control card */

Function Name: set_backlash，start_backlash，end_backlash

Purpose: Set compensation value of gaps caused by the direction turning of mechanism with 'set backlash'
'start backlash': Start compensation
'end backlash': End compensation

Syntax : int set_backlash（int ch，int backlash）;
int start_backlash（int ch）;
int end_backlash（int ch）
ch:the serial numbers of the controlling axis
backlash: Gap value caused by the direction turning, and the unit is the number of pulses;

Quoted Examples: set_backlash（1，12）;
start_backlash（1）;
end_backlash（1）

Description: Set_backlash function set a compensation value to eliminate the location error caused by the direction turning of mechanism .Quoting start_backlash function to start the reverse gap compensation to controlling axis and Quote end_backlash function to end the reverse gap compensation to controlling axis.

Return Value: If set successfully, the set_backlash,set_backlash and end _backlash function return ' 0' value, otherwise return '1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Notes: set_backlash function just set compensation value. The true compensation value dose not work until start_backlash function is quoted. Set_backlash function should be quoted before start_backlash function. Otherwise, the system will use default compensation value (default is 2 0 impulses).

Function Name: change_speed

Purpose: This function is used to achieve speed-changing in movement.

Syntax:int change_speed（int ch,double speed）;
ch: Axis number
speed: Target speed

Description: Change-speed function is used to achieve speed changing in movement. The maximum value of the speed changing should not exceed the value set by set_maxspeed function, and the minimum value must exceed '0'. The function could be used to achieve speed-changing in process of movement when the movement is started with high-speed motion command. Acceleration of speed-changing is decided by the set_profile function which quoted before motion command function.

Return Value: Return '0' when quoted correctly, or else return '1' value

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Reference: set_profile

Function Name: Outport

Purpose: Outport function is used to write a certain port address of computer

Syntax: int Output（int portid,unsigned char byte）;

int portid: Port address of computer

unsigned char byte: output a byte data

Description: The output function writes the data of a byte to the port address which corresponding to portid, such as parallel port of a computer. This function has no relations with MPC 07 card operation. It is easy to write for PC port with this function. The function must be used carefully because it can operate to any port of computer.

Return Value: Return '0' value if the function is quoted correctly, otherwise return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: Inport

Purpose: With inport function to have reading operation to port address

Syntax: int Inport（int portid）;int portid: Port address of computer

Description: Inport function read the data of the input port corresponding to address portid, such as parallel port of computer. This function has no relations with MPC 07 card operation. Reading operation of PC port can be easily done with this function. The function must be used carefully because it can operate to any port of computer.

Return Value: Return the content read

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: get_lib_ver

Purpose: Used to query the version number of function library.

Syntax: int get_lib_ver (long* major, long *minor1, long *minor2);

long * major: point to the finger of main version number of function library.

long * minor1: point to the finger of sub-version-number 1 of function library.

long * minor2: point to the finger of sub-version-number 2 of function library.

Quoted Example: get_lib_ver (&major, &minor1, &minor2);

Description: this function could be used to inquire the version number of the function library of movement controlling card. The version of function library must conform to the version of driver and board hardware.

Return Value: "0" value.

System: WINDOWS98, WINDOWS2000, WINDOWS XP

Function Name: get_sys_ver

Purpose: inquiring version number of driver.

Syntax: int get_sys_ver (long* major,long *minor1,long *minor2);

long * major: point to the finger of main version number of driver.

long * minor1: point to the finger of sub-version-number 1 of driver.

long * minor2: point to the finger of sub-version-number 2 of driver.

Quoted Example: get_sys_ver (&major, &minor1, &minor2);

Description: this function is used to inquire the version number of motion controlling card driver. Driver version must conform to the version of function library and board hardware

Return Value: if called successfully, the function return '0' value, otherwise return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP

Function Name: get_card_ver

Purpose: inquiring version number of board hardware.

Syntax: int get_card_ver (long cardno, long *type, long* major, long *minor1, long*minor2);

long cardno: Card number

long * type: point to the finger of style number of board and type number of MPC 07SP board is '0'.

long * major: point to the finger of main version of board hardware.

long * minor1: point to the finger of sub-version number 1 of board hardware

long * minor2: point to the finger of sub-version-number 2 of board hardware

Quoted Example: get_card_ver(1, &type, &major, &minor1, &minor2);

Description: this function could be used to inquire the type and version number of motion controlling card. MPC07 boards of different models have corresponding numbers.

Return Value: if quoted successfully, the function return '0'value, otherwise return '-1' value.

System: WINDOWS98, WINDOWS 2000, WINDOWS XP